SISTEMI OPERATIVI

ESERCIZIO N. 1 del 9 DICEMBRE 2002

In una cooperativa, **A** artigiani condividono **S** strumenti numerati da 1 a S che usano per il loro lavoro. Ogni artigiano, per lavorare, ha sempre bisogno di 3 strumenti specifici s1, s2 e s3 (tali valori numerici dipendono dal lavoro da svolgere), che richiede e prende se sono liberi, e poi restituisce alla fine del lavoro. Gli artigiani sono di due tipi: **specializzati** o **semplici**. I primi hanno priorità sui secondi.
Si implementi una soluzione usando il costrutto monitor per modellare la **cooperativa** e i processi per modellare gli **artigiani** e si descriva la sincronizzazione tra i processi. Nel rispettare i vincoli richiesti, si cerchi di massimizzare l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

```
program Cooperativa

const     S = ...; { numero di strumenti }
type      strum = 1..S;
type      tipo = (spec, semp);

type artigiano = process (t: tipo)
begin
      repeat
            < decidi s1, s2, s3 >
            c.richiedi (t, s1, s2, s3);
            <fa il suo lavoro >
            c.rilascia (s1, s2, s3);
      until false
end

type cooperativa = monitor
{ variabili del monitor }
var  sospesi: array[tipo] of integer;
      { numero di artigiani sospesi }
      coda : array[tipo] of condition;
      { code su cui sospendere gli artigiani }
      occupato : array[strum] of boolean;
      { dice se gli strumenti sono occupati }

procedure entry richiedi (t: tipo; s1, s2, s3: strum)
begin
      { se c'è almeno uno strumento occupato }
      while (occupato[s1] or occupato[s2] or occupato[s2] or
        { o c'è uno specializzato in coda }
        (t = semp and coda[spec].queue) do
      begin
            sospesi[t] ++;
            coda[t].wait;
            sospesi[t] --;
      end
```

```
        { acquisisce la risorsa }
        occupato[s1] := true
        occupato[s2] := true
        occupato[s3] := true
end

procedure entry rilascia (s1, s2, s3: strum)
var s, i: integer;
begin
        { rilascia le risorse }
        occupato [s1] := false ;
        occupato [s2] := false ;
        occupato [s3] := false ;

        { risveglia tutti gli artigiani }
        s := sospesi[spec];
        for i := 1 to s do
                coda[spec].signal;
        s := sospesi[semp];
        for i := 1 to s do
                coda[semp].signal;
end

begin { inizializzazione delle variabili }
        sospesi[spec] := 0;
        sospesi[semp] := 0;
        for i := 1 to S do
                occupato [i] := false;
end

var  c: cooperativa; { il nostro monitor }
        a1, a2, ..., aA : artigiano (t, j, k, l);

begin end.
```

Starvation

La soluzione proposta presenta starvation nei confronti degli artigiani semplici, i quali possono essere scavalcati in modo indefinito dagli artigiani specializzati.
Per evitare ciò, si può imporre di alternare la priorità ogni tot di esecuzioni, tenendone conto tramite un contatore.

NOTE

Poiché gli artigiani hanno bisogno di tre strumenti, si è preferito sospenderli tutti in una unica coda, risvegliarli tutti quando un artigliano libera gli strumenti che ha usato, e lasciare che siano essi stessi a ritestare le condizioni in un ciclo while.

2° soluzione che sfrutta meglio le risorse

```
program Cooperativa

const     S = ...; { numero di strumenti }
type      strum = 1..S;
type      tipo = (spec, semp);

type artigiano = process (t: tipo)
{ come prima }

type cooperativa = monitor

{ variabili del monitor: come prima più }
var  SpecAtt : array[strum] of integer;
     { specializzati in attesa di uno strumento }

procedure entry richiedi (t: tipo; s1, s2, s3: strum)
begin
     if (t = semp)
     begin
     { se c'è almeno uno strumento occupato }
     while (occupato[s1] or occupato[s2] or occupato[s2] or
      { o c'è uno specializzato che aspetta gli strumenti }
       SpecAtt[s1] > 0 or SpecAtt[s2] > 0
       or SpecAtt[s3] > 0 do
          begin
               sospesi[t] ++;
               coda[t].wait;
               sospesi[t] --;
          end
     end
```

```
        else { t = spec }
        begin
        { se c'è almeno uno strumento occupato }
        while (occupato[s1] or occupato[s2] or occupato[s2] do
            begin
                sospesi[t] ++;
                SpecAtt[s1] ++;
                SpecAtt[s2] ++;
                SpecAtt[s3] ++;
                coda[t].wait;
                SpecAtt[s1] --;
                SpecAtt[s2] --;
                SpecAtt[s3] --;
                sospesi[t] --;
            end
        end

        { acquisisce la risorsa }
        occupato[s1] := true
        occupato[s2] := true
        occupato[s3] := true
end

procedure entry rilascia (s1, s2, s3: strum) { come prima }

begin { inizializzazione delle variabili come prima più }
        for i := 1 to S do
            SpecAtt[i] := 0;
end

var  { come prima }
begin end.
```